Active Learning of Markov Decision Processes for System Verification

Chen, Yingke; Nielsen, Thomas Dyhre

Published in:
International Conference on Machine Learning and Applications (ICMLA)

DOI (link to publication from Publisher):
10.1109/ICMLA.2012.158

Publication date:
2012

Document Version
Early version, also known as pre-print

Active Learning of Markov Decision Processes for System Verification

Yingke Chen and Thomas Dyhre Nielsen
Department of Computer Science, Aalborg University
Aalborg, Denmark
{ykchen, tdn}@cs.aau.com

***Abstract*—Formal model verification has proven a powerful tool for verifying and validating the properties of a system. Central to this class of techniques is the construction of an accurate formal model for the system being investigated. Unfortunately, manual construction of such models can be a resource demanding process, and this shortcoming has motivated the development of algorithms for automatically learning system models from observed system behaviors. Recently, algorithms have been proposed for learning Markov decision process representations of reactive systems based on alternating sequences of input/output observations. While alleviating the problem of manually constructing a system model, the collection/generation of observed system behaviors can also prove demanding. Consequently we seek to minimize the amount of data required. In this paper we propose an algorithm for learning deterministic Markov decision processes from data by actively guiding the selection of input actions. The algorithm is empirically analyzed by learning system models of slot machines, and it is demonstrated that the proposed active learning procedure can significantly reduce the amount of data required to obtain accurate system models.**

***Index Terms*—Active learning; verification; Markov decision processes; statistical learning;**

I. Introduction

Model checking is becoming increasingly popular for validating and verifying the behavior of a software system by comparing a formal model of the system with its specification or intended behavior. Unfortunately, establishing an accurate formal model representation of a complex software system can be a tedious and time consuming process that is often seen as a hindrance for the practical deployment of otherwise powerful formal model-based verification techniques. This is e.g. the case for systems consisting of multiple embedded and interacting software components, some of which may rely on 3rd party systems with limited detailed and up-to-date documentation. In order to address this issue, methods have been proposed for automatically *learning* a formal system model based on observations of the (black-box) system under consideration. After a formal model has been established, existing model checking procedures and other model driven development techniques can then be deployed using the learned model.

Learning non-probabilistic systems has been addressed in, e.g. [1]. However, non-probabilistic systems are often not sufficiently flexible for modeling the behavior of complex software systems whose behavior are influenced by e.g. unpredictable user inputs, interactions with the surrounding environment, and randomized algorithms. To alleviate these shortcomings, methods for learning probabilistic system models have recently been developed [2], [3], [4]. In particular, based on the Alergia algorithm [5] for learning probabilistic finite automata, [6] proposed the IOAlergia algorithm for learning deterministic Markov decision processes (MDPs) based on data consisting of alternating input/output symbols. These types of models are used for modeling reactive systems, where the input actions are chosen non-deterministically (representing e.g. user interactions or interfaces to other components) and the output is determined probabilistically conditioned on the input [7]. By using MDPs we can establish models of individual system components and model by non-determinism the interaction with the environment and the other components of the system. Model checking MDPs therefore involve analyzing the behavior of the model under all possible interactions with the environment.

Convergence results of the learning method by [6] guarantee that, in the large sample limit, IOAlergia will identify the generating model up to bisimulation equivalence. For practical applications, however, the available data is often limited as the generation/collection of large amounts of data can be a resource demanding task. Thus, it is important to find methods for reducing the amount of data required for establishing an accurate system model. One possible approach to reduce the amount of data is to deploy *active learning* [8]. Rather than selecting input actions randomly (as done in [6]), active learning seeks to guide the selection of input actions in a direction that is more likely to lead to accurate system models.

Active learning has previously been explored in the machine learning community [9], [10] as well as in the verification and model checking community. In the model checking community, active learning has mainly been centered around non-probabilistic models. Examples include membership and equivalence queries [11] as well as co-evolutionary learning [12] of deterministic finite automata (DFA). In the latter approach, a set of candidate models are maintained, and a new training instance is selected so as to cause maximal disagreement among these models. In this paper we propose an active learning method based on the IOAlergia algorithm for Markov decision processes. The performance of the algorithm is analyzed by actively learning an MDP model for a slot machine. The analysis is conducted by comparing the amount of data required for learning an accurate system model compared to the passive learning approach employed by the original IOAlergia algorithm. The model comparison is conducted by analyzing the linear temporal logic properties of the learned models as well as the maximum expect rewards that can be obtained from the models.

The remainder of this paper is organized as follows: Section II contains a formal specification of the background material required for the learning algorithm; Section III outlines the existing IOALERGIA algorithm, Section IV describes the active learning framework, and section V presents an experimental study of the algorithm. Finally, section VI concludes the paper.

II. PRELIMINARY

A. Markov decision processes

The learning algorithm proposed in this paper is based on labeled Markov decision processes.

Definition 1 (LMDP)**.** *A* Labeled Markov Decision Processes (LMDP) *is a tuple* $M = (Q, \Sigma_I, \Sigma_O, \pi, \tau, L)$

- Q *is a finite set of states,*
- Σ_I *is a finite input alphabet, and* Σ_O *is a finite output alphabet,*
- $\pi : Q \to [0,1]$ *is an initial probability distribution such that* $\sum_{q \in Q} \pi(q) = 1$*,*
- $\tau : Q \times \Sigma_I \times Q \to [0,1]$ *is the transition probability function such that for all* $q \in Q$ *and all* $\alpha \in \Sigma_I$*,* $\sum_{q' \in Q} \tau(q, \alpha, q') = 1$*, or* $\sum_{q' \in Q} \tau(q, \alpha, q') = 0$
- $L : Q \to \Sigma_O$ *is a labeling function.*

For a given set of atomic propositions AP, we define that $\Sigma_O = 2^{AP}$. An input $\alpha \in \Sigma_I$ is said to be enabled in state $q \in Q$ if and only if $\sum_{q' \in Q} \tau(q, \alpha, q') = 1$, and the set of enabled input actions in state q is denoted $Act(q)$.

An LMDP is said to be *deterministic* if it contains a unique starting state and if, for each state and input action, the possible successor states are uniquely labeled. More formally:

Definition 2 (DLMDP)**.** *A LMDP is deterministic if*

- *There exists a state* $\mathrm{q}_s \in Q$ *with* $\pi(\mathrm{q}_s) = 1$*,*
- *For all* $q \in Q$*,* $\alpha \in \Sigma_I$ *and* $\sigma \in \Sigma_O$*, there exists at most one* $q' \in Q$ *with* $L(q') = \sigma$ *and* $\tau(q, \alpha, q') > 0$*. We then also write* $\tau(q, i, \sigma)$ *instead of* $\tau(q, i, q')$*.*

A labeled MDP M can be used for modeling reactive systems, where the system makes a probabilistic move based on the current state and a chosen input action. In order to specify probabilities over events of M, we need to resolve the non-probabilistic decisions in the model: A *scheduler* $\mathfrak{S}$ for an MDP M is a function that in any state q chooses an action $\alpha \in \Sigma_I$, i.e, $\mathfrak{S} : Q^+ \to \Sigma_I$ such that $\mathfrak{S}(q_0 q_1 \dots q_n) \in \Sigma_I$, for all $q_0, q_1, \dots, q_n \in Q^+$. For an MDP M, a scheduler $\mathfrak{S}$ thus resolves the non-probabilistic choices of M, thereby turning M into a Markov chain $M_{\mathfrak{S}}$ [13, Section 10.6].

A labeled Markov chain (LMC) $M_{\mathfrak{S}}$ induced by an LMDP M and a scheduler $\mathfrak{S}$ defines a probability measure $P_{M_{\mathfrak{S}}}$ on $(\Sigma_O)^\omega$ which is the basis for associating probabilities with events in the LMC $M_{\mathfrak{S}}$: the probability of a string $s = \sigma_0 \sigma_1 \dots \sigma_n, \sigma \in \Sigma_O$ is given by:

$$P_{M_{\mathfrak{S}}}(s) = \prod_{i=1}^{n} \tau_{\mathfrak{S}}(\sigma_0 \sigma_1 \dots \sigma_{i-1}, \sigma_i).$$

B. Probabilistic LTL

Linear time temporal logic (LTL) [14] was proposed for verifying the correctness of systems based on execution sequences, what property specified by an LTL formula need not only depend on the current state but can also relate to future states.

Linear time temporal logic (LTL) is defined by the syntax

$$\varphi ::= a \mid \varphi_1 \wedge \varphi_2 \mid \neg\varphi \mid \bigcirc \varphi \mid \varphi_1 \mathsf{U} \varphi_2 \qquad a \in \Sigma_O.$$

For better readability, we also use the derived temporal operators $\Box$ (always) and $\Diamond$ (eventually).

Let φ be an LTL formula. For $s = \sigma_0 \sigma_1 \dots \in (\Sigma_O)^\omega$, $s[j \dots] = \sigma_j \sigma_{j+1} \sigma_{j+2} \dots$ is the suffix of s starting with the (j)st symbol σ_j. The LTL semantics for infinite words over Σ are given as follows:

- $s \models true$
- $s \models \sigma$, iff $\sigma = \sigma_0$
- $s \models \varphi_1 \wedge \varphi_2$, iff $s \models \varphi_1$ and $s \models \varphi_2$
- $s \models \neg \varphi$, iff $s \not\models \varphi$
- $s \models \bigcirc \varphi$, iff $s[1 \dots] \models \varphi$
- $s \models \varphi_1 \mathsf{U} \varphi_2$, iff $\exists j \geq 0.\ s[j \dots] \models \varphi_2$ and $s[i \dots] \models \varphi_1$, for all $0 \leq i < j$

The syntax of probabilistic LTL (PLTL) is:

$$\phi ::= P_{\bowtie r}(\varphi) \quad (\bowtie \in \geq, \leq, =;\ r \in [0,1];\ \varphi \in \text{LTL}).$$

A labeled Markov decision process M satisfies the PLTL formula $P_{\bowtie r}(\varphi)$ iff $P_{M_{\mathfrak{S}}}(\varphi) \bowtie r$ for all schedulers of M, where $P_{M_{\mathfrak{S}}}$ is the probability distribution defined by the LMC induced by the scheduler $\mathfrak{S}$ of M, and $P_{M_{\mathfrak{S}}}(\varphi)$ is short for $P_{M_{\mathfrak{S}}}(s | s \models \varphi, s \in (\Sigma_O)^\omega)$.

The quantitative analysis of an MDP M against specification φ amounts to establishing the lower and upper probability bounds that can be guaranteed when ranging over all possible schedulers. This corresponds to computing

$$P_M^{\max}(\varphi) = \sup_{\mathfrak{S}} P_{M_{\mathfrak{S}}}(\varphi) \ \text{ and } \ P_M^{\min}(\varphi) = \inf_{\mathfrak{S}} P_{M_{\mathfrak{S}}}(\varphi),$$

where the infimum and the supremum are taken over all schedulers for M.

III. LEARNING DETERMINISTIC LABELED MDPS

In this section we will briefly describe the IOALERGIA algorithm presented in [6] and which form the basis for the active learning algorithm proposed in the paper.

A. Passive data generation

The data used by the IOALERGIA algorithm is assumed to consist of multiple sequences of alternating input/output symbols, where each sequence is produced by observing the behavior of a system modeled by an DLMDP M: the data generation is initiated by observing the label of the initial state of M. At each point in time thereafter an input action $\alpha \in \Sigma_I$ is randomly chosen (independently of the current state) causing the system to either i) make a probabilistic transition to a successor state for which the state label is observed if α is enabled or ii) stay in the current state and

output a special error symbol *err* if α is not enabled. Using this data generation procedure, an observed data sequence will consist of an initial output symbol followed a sequence of input/output pairs $\sigma_0\alpha_1\sigma_1 \ldots \alpha_n\sigma_n$, where $\alpha_i \in \Sigma_I, \sigma_j \in \Sigma_O \cup \{err\}$. The length n of the sequence is assumed to be randomly determined according to a geometric distribution and the sequences in the data set are assumed to be independent.

For some models, there may exist a uniquely labeled absorbing state, which can be identified by its state label (e.g., a failure state from which the system cannot recover). When prior knowledge is available, observations can be stopped when such a state is encountered.

B. The IOAlergia algorithm

Given a dataset S generated according to the above procedure, the IOALERGIA algorithm proceeds in two steps. Firstly, a so-called *input/output frequency prefix tree acceptor* (IOFPTA)[1] is constructed as a representation of the data. An IOFPTA T is a directed tree structure, where each node is labeled by an output symbol and each link is labeled by an input symbol. A path from the root q_r of T to a node q_i corresponds to a prefix of a string in S defined by the labels on the nodes and the links on the path.[2] Each node s in the tree is furthermore annotated with a *transition frequency function* $f(q, \alpha, \sigma)$ ($\alpha \in \Sigma_I,\ \sigma \in \Sigma_O$) which is the number of strings in S with prefix $q\alpha\sigma$ and $f(q, \alpha) = \sum_{\sigma\in\Sigma_O} f(q, \alpha, \sigma)$. Given an input action and an output symbol at a node in an IOFPTA, the next state/node can be uniquely determined. An IOFPTA can be transformed to a DLMDP by normalizing the frequencies $f(s, \alpha, \cdot)$ to $\tau(s, \alpha, \cdot)$.

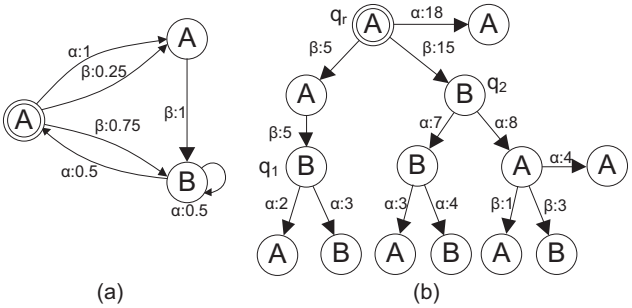

Fig. 1. (a) A DLMDP over $\Sigma_O = \{A, B\}$ and $\Sigma_I = \{\alpha, \beta\}$; (b) An IOFPTA representation of data generated from the DLMDP in (a).

Example 1. *The* IOFPTA *in Fig. 1(b) is constructed from sample sequences generated by the DLMDP M in Fig. 1(a). The root node (double circled) is labeled by A. From the root node, given different input actions α and β, different successor nodes can be reached by strings with the prefixes $A\alpha A$ $A\alpha B$, and $A\beta A$. For the root node q_r we,e.g., have the frequencies $f_{q_r}(A, \beta, A) = 5$ and $f_{q_r}(A, \beta, B) = 15$, which specified that there are 5 and 15 sequences with prefix $A\alpha A$ and $A\alpha B$, respectively.*

[1]The term is adopted from [5] on which the algorithm is based.

[2]Since there is a one-to-one correspondence between a node q in T and a string s in S, we will sometimes use the terms interchangeably.

Given an IOFPTA representation of the data, the IOALERGIA algorithm proceeds by iteratively merging nodes that pass a compatibility test. Intuitively, two nodes are merged if they can be mapped to the same state in the generating model. More formally, two nodes, q_1 and q_2 are said to be *ϵ-compatible*, if it holds that:

1) $L(q_1) = L(q_2)$;
2) For all $\alpha \in \Sigma_I$ and $\sigma \in \Sigma_O$, $\left|\frac{f(q_1,\alpha,\sigma)}{f(q_1,\alpha)} - \frac{f(q_2,\alpha,\sigma)}{f(q_2,\alpha}\right| < (\sqrt{\frac{1}{f(q_1,\alpha)}} + \sqrt{\frac{1}{f(q_2,\alpha)}}) \cdot \sqrt{\frac{1}{2} \cdot \ln\frac{2}{\varepsilon}}$;
3) The successor nodes $q_1\alpha\sigma$ and $q_2\alpha\sigma$ of q_1 and q_2 are *ϵ-compatible*, for all $\alpha \in \Sigma_I$, and $\sigma \in \Sigma_O$.

Condition 1) requires that the two nodes have the same label. Condition 2), also known as Hoeffding test [15], bounds the allowed difference between the distributions of the two nodes also taking the uncertainty about the distributions into account. The last condition requires the compatibility to be recursively satisfied for every pair of successors states. As an example, consider the IOFPTA in Fig. 1(b), where the states q_1 and q_2 pass the compatibility test and are therefore be merged. The merge procedure is illustrated in the example below.

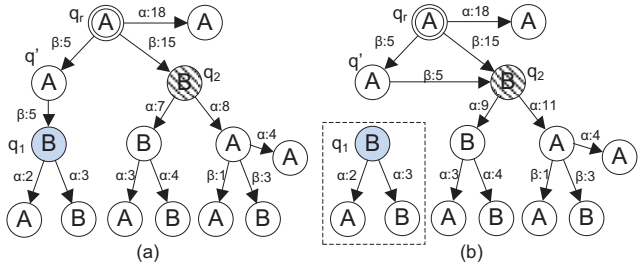

Fig. 2. The figure shows the process of merging states q_1 and q_2.

Example 2. *Fig. 2 shows the merge procedure where node q_1 (shadowed) is merged with the node q_2 (dashed). Firstly, the transition from node q' to q_1 is redirected to q_2. Then, transitions from q_1 to its successor nodes are folded into the corresponding successor nodes of q_2 and the frequencies are updated. Finally, the nodes in the dashed box are removed from the model.*

IV. ACTIVE LEARNING

The IOALERGIA algorithm is based on an IOFPTA representation T of the data. The basic idea of the algorithm is to find an approximation of the generating model M by grouping together states in T, which can be mapped to a single state in M. For two nodes to be grouped together they should be compatible. To test whether states are compatible the Hoeffding test is deployed, which, roughly speaking, measures the difference in the probability distributions defined at the two states also taking our confidence about the distributions into account (i.e, the amount of data used for estimating the distributions).

Intuitively, in order to adapt the IOALERGIA algorithm to an active learning setting we aim at (indirectly) reducing the number of false compatibility tests by generating data sequences minimizing the uncertainty about the IOFPTA representation T, which, in turn, implies reducing the uncertainty about the derived transition probabilities in T.[3]

For each state q in a DLMDP, the transition probabilities $P(\Sigma_O|q,i,\boldsymbol{\theta}^{(q,i)})$ conditioned on a specific input action i follow a multinomial distribution and is parameterized by $\boldsymbol{\theta}^{(q,i)} = (\theta_1^{(q,i)},...,\theta_{|\Sigma_O|}^{(q,i)})$. In order to express our uncertainty about $P(\Sigma_O|q,i,\boldsymbol{\theta}^{(q,i)})$ we specify a density $P(\boldsymbol{\theta}^{(q,i)})$ over the possible parameter values $\boldsymbol{\theta}^{(q,i)}$. By making the standard assumption of *parameter independence* we can specify the joint density $P(\boldsymbol{\theta})$ over all parameters $\boldsymbol{\theta}$ in the model by a collection of local parameter densities. An appropriate prior distribution over the parameters is the Dirichlet distribution, which is the conjugate prior distribution for the multinomial distribution [16] and is parameterized by the hyperparameters $(\alpha_1^{(q,i)},...,\alpha_{|\Sigma_O|}^{(q,i)})$, $\alpha_j^{(q,i)} \in \mathbb{R}^+$. The hyperparameter $\alpha_j^{(q,i)}$ can be thought of as representing the number of times we have observed the jth output symbol in state q after input action i. If $P(\Sigma_O|q,i)$ is parameterized by $\boldsymbol{\theta}^{(q,i)}$, and $p(\boldsymbol{\theta}^{(q,i)})$ follows a Dirichlet distribution, then the probability of observing $o_j \in \Sigma_O$ can be estimated as α_j/α_*, where $\alpha_* = \sum_{j=1}^{|\Sigma_O|}\alpha_j^{(q,i)}$. Furthermore, if q_k is observed after action i in state q, then the posterior distribution $P(\boldsymbol{\theta}^{(q,i)})$ is still Dirichlet distributed with hyperparameters $(\alpha_1^{(q,i)},\ldots,\alpha_k^{(q,i)}+1,\ldots,\alpha_{|\Sigma_O|}^{(q,i)})$.

Since the actively learning procedure will incrementally generate new data, the IOFPTA representation of the data is not fixed but rather grows as more data is added. Thus, we may think of the tree as consisting of an observed initial part (corresponding to the data) as well as an unobserved/virtual part. Special attention must be given to the unobserved part of the T, i.e., the part of the IOFPTA for which no observations have (yet) been made. For the nodes in this part of the tree we shall a priori assume a uniform distribution over Σ_O, where the uncertainty about the parameter values is modeled by a Dirichlet distribution with hyperparameters $(1_1,1_2,\ldots,1_{|\Sigma_O|})$.

With the specification above, our active learning algorithm can now be stated as follows: on each state q with $P(\boldsymbol{\theta}^{(q,\cdot)})$, select an action $i \in \Sigma_I$ and observe the resulting output symbol $o_j \in \Sigma_O$. Based on the observation o_j, update the distribution $P(\boldsymbol{\theta}^{(q,i)})$ to obtain the posterior distribution $P'(\boldsymbol{\theta}^{(q,i)})$. In what follows we will derive a measure $\mathsf{G}(q,i)$ for estimating the value of the different actions i in any given state q.

The distribution $P(\boldsymbol{\theta})$ represents our current knowledge about the parameters in T. When using the IOALERGIA algorithm, we need to select a set of specific parameter values, and for that we use the expected value of $\boldsymbol{\theta}$, which we denote by $\tilde{\boldsymbol{\theta}}$. However, if the "true" parameters are $\boldsymbol{\theta}^*$ rather than $\tilde{\boldsymbol{\theta}}$ we will incur a *loss* $\mathsf{Loss}(\tilde{\boldsymbol{\theta}}||\boldsymbol{\theta}^*)$ (to be defined below). The goal is to minimize the loss through active learning, but since we do not know $\boldsymbol{\theta}^*$ we will instead aim at minimizing the *expected loss* (or *risk*):

$$\begin{aligned}\mathsf{Risk}(P(\boldsymbol{\theta})) &= E_{\Theta\sim P(\boldsymbol{\theta})}\mathsf{Loss}(\Theta||\tilde{\boldsymbol{\theta}}) \\ &= \int_{\boldsymbol{\theta}} \mathsf{Loss}(\boldsymbol{\theta}||\tilde{\boldsymbol{\theta}})P(\boldsymbol{\theta})d\boldsymbol{\theta},\end{aligned}$$

where the expectation is taken wrt. our current belief $P(\boldsymbol{\theta})$ about the value of $\boldsymbol{\theta}^*$. The risk of $P(\boldsymbol{\theta})$ can therefore be seen as a measure of the quality of our model.

If $P(\boldsymbol{\theta}|q,i,o)$ is the posterior distribution over the parameters given a new pair of input and output symbols in state q, then we define the *expected posterior risk of input action* i as

$$\mathsf{ERisk}(P(\boldsymbol{\theta}|q,i)) = E_{\boldsymbol{\Theta}\sim P(\boldsymbol{\theta})}E_{O\sim P(\Sigma_O|q,i,\boldsymbol{\Theta}))}\mathsf{Risk}(P(\boldsymbol{\theta}|q,i,O)),$$

When evaluating an input action i in state q an immediate approach is therefore to consider the expected reduction risk:

$$\Delta_{q,i} = \mathsf{ERisk}(P(\boldsymbol{\theta}|q,i)) - \mathsf{Risk}(P(\boldsymbol{\theta})).$$

However, the input action i at state q not only reduces the local uncertainty about the distribution over Σ_O, but it also influences which states can be visited next and therefore the potential reduction in uncertainty of future states. Thus, the expected uncertainty reduction of the possible successor states should be taken into account as well. With the assumption that the length of the sample sequences follow a geometric distribution (see Section III-A), we have an identical termination probability at every state. In other words, there is fixed probability γ that the sample generation will terminate at the next state, and we therefore define the *gain* of the input action i at state q as:

$$\mathsf{G}(q,i) = \sum_{o_j\in\Sigma_O} P(o_j|q,i)[\Delta_{q,i,o_j} + \gamma \cdot \underset{k}{max}(\mathsf{G}(qio_j,k))],$$

where $\Delta_{q,i,o_j} = \mathsf{Risk}(P(\boldsymbol{\theta}|q,i,o_j)) - \mathsf{Risk}(P(\boldsymbol{\theta}))$ is the *local risk reduction* and qio_j is the (unique) state that is reached from q by input i and output o_j. Thus, $\mathsf{G}(q,i)$ combines the immediate reduction in expected posterior risk with the maximum expected posterior risk reduction at future states (weighted with the probability of reaching these states). Observe that γ also serves as a discounting factor ensuring that the gain is always finite (assuming that the expected posterior risk is finite).

Before specifying a bound for the recursion above, we first need to consider a suitable loss function. For that we follow [17] and define the loss $\mathsf{Loss}(\tilde{\boldsymbol{\theta}}||\boldsymbol{\theta})$ as the Kullback-Leibler divergence (KL) between the two distributions induced by $\tilde{\boldsymbol{\theta}}$ and $\boldsymbol{\theta}$:

$$\mathsf{KL}(\boldsymbol{\theta},\tilde{\boldsymbol{\theta}}) = \Sigma_{o\in\Sigma_O}P(o|\boldsymbol{\theta})\ln\left(\frac{P(o|\boldsymbol{\theta})}{P(o|\tilde{\boldsymbol{\theta}})}\right)$$

[3]The transition probabilities are obtained by normalizing the frequencies of T thereby producing a DLMDP. In what follows we will, unless states otherwise, also consider IOFPTAs with normalized frequencies.

With $\mathsf{KL}(\boldsymbol{\theta}, \tilde{\boldsymbol{\theta}}) = \mathsf{Loss}(\tilde{\boldsymbol{\theta}}, \boldsymbol{\theta})$ the local risk reduction Δ_{q,i,o_j} can be calculated as

$$\begin{aligned}\Delta_{q,i,o_j} &= \mathsf{Risk}(P(\boldsymbol{\theta})|q,i,o_j) - \mathsf{Risk}(P(\boldsymbol{\theta})) \\ &= \mathsf{H}\left(\frac{\alpha_{o_1}}{\alpha_{o_*}}, \ldots, \frac{\alpha_{o_n}}{\alpha_{o_*}}\right) - \mathsf{H}\left(\frac{\alpha'_{o_1}}{\alpha'_{o_*}}, \ldots, \frac{\alpha'_{o_n}}{\alpha'_{o_*}}\right),\end{aligned}$$

where H is the entropy function, $\alpha_{o_*} = \sum_{o_i \in \Sigma_O} \alpha_{o_i}$, and $n = |\Sigma_O|$. Furthermore, $\alpha'_{o_i} = \alpha_{o_i} + 1$ if $o_i = o$ and $\alpha'_{o_i} = \alpha_{o_i}$ otherwise; $\alpha'_{o_*} = \alpha_{o_*} + 1$. In particular, for any non-visited state q we have that $\Delta_{q,i,o} = \Delta_* = \mathsf{H}(\frac{[1,\ldots,1]}{n}) - H(\frac{[1,\ldots,2,\ldots,1]}{n+1})$ for all $i \in \Sigma_I$, and the gain of any non-visited state is therefore

$$\mathsf{G}(q,i) = \frac{1}{1-\gamma}\Delta_*,$$

which also terminates the recursive definition of $\mathsf{G}(q,i)$. The proof for the calculation above follow that given in [17] for regular Bayesian network models.

In summary, the active learning procedure is initialized with a virtual infinite IOFPTA with no symbols observed and with uniform prior distributions over the output symbols. At each state q, the input action i with highest gain $\mathsf{G}(q,i)$ is chosen and the associated distribution over $\boldsymbol{\theta}^{(q,i)}$ is updated based on the output symbol observed.

As a final comment concerning complexity we note that when updating the distributions in the IOFPTA during active learning, only the distributions associated with the nodes that have been visited are updated. This also means that when recalculating the gain $\mathsf{G}(q,\cdot)$ (after having completed an iteration and returning to the initial state) we only need to update the calculations in the part of the tree pertaining to the nodes that were visited during the last iteration.

V. Experimental Results

In this section, we are going to analyze the active learning algorithm using a case study based on the slot machine model originally presented in [18] and adapted in [6]. The slot machine consists of 3 reels, where each reel has 4 symbols. By paying one coin each real can be spun once, and for each extra coin inserted the player gets one extra spin on any of the three reels. The player can stop and collect a prize if every reel has been spun at least once and if the current configuration of the reels generates a prize. The prizes for the various reel configurations are listed in Table I and the probability distribution over the four symbols on the reels are listed in Table II.

The active learning algorithm discussed previously has been implemented to choose the input action (i.e., which of the three reels to spin) at each step during the data collection procedure. Based on the "true" slot machine model we have generated datasets of different sizes using both active and passive data generation. IOALERGIA was then applied for learning a DLMDP based on the different datasets; the data was also used to estimate the discounting factor γ. For all data sets, we assume access to prior knowledge about the domain,

TABLE I
SLOT MACHINE PRIZES.

r_1	r_2	r_3	Prz
bar	bar	bar	10
cherry	cherry	cherry	10
grapes	grapes	grape	10
?	bar	bar	5
cherry	?	cherry	5
grapes	grapes	?	5
?	?	bar	2
?	?	cherry	1

TABLE II
PROBABILITY OF SYMBOLS FOR THE THREE REELS.

	r_1	r_2	r_3
lemon	0.25	0.4	0.4
grape	0.25	0.1	0.2
cherry	0.25	0.2	0.2
bar	0.25	0.3	0.2

which is simulated by initially generating (passively) a data set with 1600 observation sequences. The prior knowledge is used to provide the initial guidance to the active learner.

The learned models have been evaluated using three criteria. First, we have calculated the log-likelihood of an independent test set consisting of 300 sample executions. The result of this experiment is shown in Fig. 3(a) for a slot machine allowing 4 spins in total. Observe that the log-likelihood of the test data for the model learned using active learning is consistently higher than the one learned from passively generated data. The error bars are based on 10 iterations of data generation.

Secondly, we have considered the maximum and minimum probabilities of eventually getting the different prizes listed in Table I. These probabilities can be specified by the PLTL formulas $P^{\max}(\Diamond \mathrm{L}\ coins)$ and $P^{\min}(\Diamond \mathrm{L}\ coins)$, where $\mathrm{L} \in \{0,1,2,5,10\}$. The results are summarized in Fig. 3(b), for a slot machine allowing 4 spins, and shows the mean absolute difference of these PLTL properties between the generating model and the learned models. From the figure we see that the model found using active learning has smaller mean absolute difference compared to the one obtained using passively generated data. The probabilities have been calculated using the probabilistic model checker PRISM [19].

Thirdly, we have compared the learning approaches based on the maximum expected reward specified by the generating model and the models found using actively and passively generated data. This property can be specified as $R^{\max}(\Diamond stop)$ using PRISM notation, which denotes the maximal expected reward of reaching the termination state in one gamble. As shown in Fig. 3(c), both learning approaches overestimate the expected reward, but the estimated values approaches the true value as the amount of data increases and by using actively generated data generally produce better results.

As a brief summary, Fig. 3 shows that the active learning approach provides a better approximation than passive learning in three different aspects. Another interpretation is that active learning can obtaining the same level of accuracy (e.g. in terms of PLTL properties) with less data. For example, in Fig. 3(a) we see that models learned from 3000 actively generate sequences provide the same level of accuracy as models learned from 6400 passively generated data sequences.

In Fig. 4, a comparison of the amount of data required to achieve similar performance for active and passive learning is conducted. Each plot in the figure corresponds to one of the three criteria above, and each point in a plot shows the

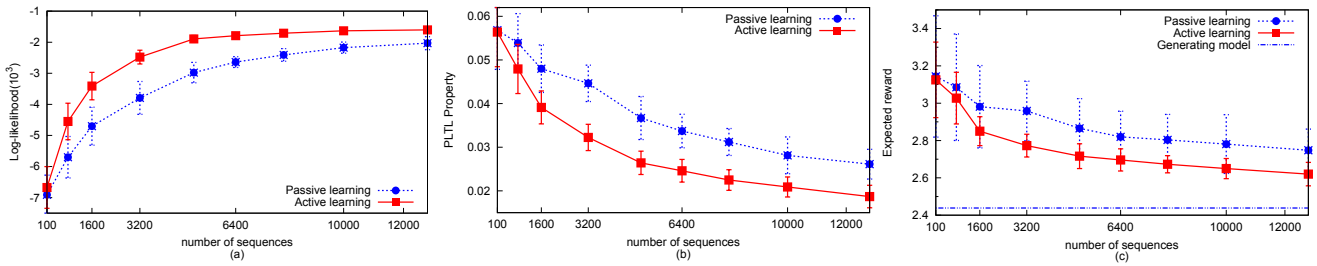

Fig. 3. The figures show learning results for a slot machine with 4 spin chances and with 1600 passively generated sequences as prior knowledge. (a) Log-likelihood of 300 passively generated sample executions as test data; (b) The absolute mean difference of a set of PLTL formulas between generating models and learned models. (c) The maximal expected reward of both the generating and learned model. The errorbars are formed from 10 iterations of the experiments.

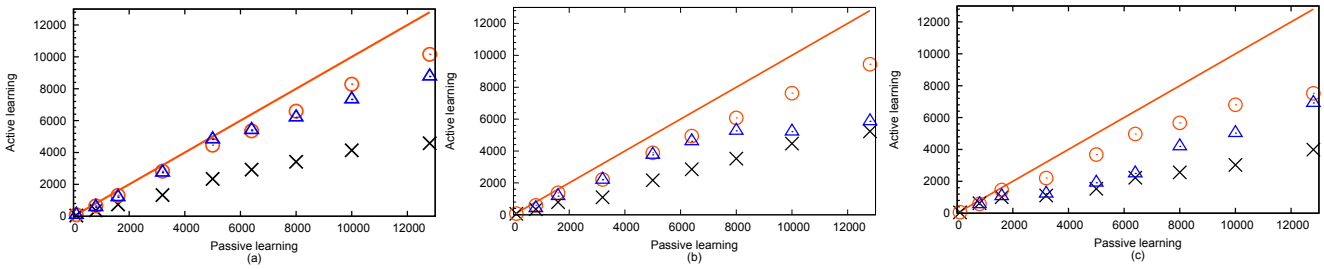

Fig. 4. Comparison between active learning and passive learning in terms of required data to achieve similar performance. A cross, triangle and circle corresponds to a slot machine with 4, 6, and 8 spin chances, respectively. The performances are measured by (a) log-likelihood of independent test data, (b) preservation of PLTL properties, and (c) maximal expected reward of each gamble.

amount of data required by passive and active learning to achieve similar performance for slot machines allowing 4, 6, and 8 spins. We observe that actively learning consistently outperforms passively learning using also these slot machines.

VI. Conclusion

In this paper, we have proposed an active learning algorithm for learning deterministic labeled Markov decision processes (DLMDPs) for formal model checking. The learning algorithm is based on data in the form of observed input/output behavior of the system to be checked and relies on the IOALERGIA algorithm [6]. The proposed algorithm attempts to guide the generation of input actions in a direction that reduces the discounted uncertainty about the local probability distributions in the model. The algorithm is empirically analyzed using a case study on slot machines. The difference between the proposed active learning algorithm and the existing passive learning procedure is compared in terms of their model properties and the maximum expected rewards that they can return. The experimental results show that the active learning algorithm can significantly reduce the amount of data required for learning accurate system models.

References

[1] F. Aarts and F. W. Vaandrager, "Learning I/O automata," in *CONCUR*, pp. 71–85, 2010.

[2] K. Sen, M. Viswanathan, and G. Agha, "Learning continuous time Markov chains from sample executions," in *QEST*, pp. 146–155, 2004.

[3] H. Mao, Y. Chen, M. Jaeger, T. D. Nielsen, K. G. Larsen, and B. Nielsen, "Learning probabilistic automata for model checking," in *QEST*, pp. 111–120, 2011.

[4] Y. Chen, H. Mao, M. Jaeger, T. D. Nielsen, K. G. Larsen, and B. Nielsen, "Learning Markov models for stationary system behaviors," in *NFM*, pp. 216–230, 2012.

[5] R. C. Carrasco and J. Oncina, "Learning stochastic regular grammars by means of a state merging method," in *ICGI*, pp. 139–152, 1994.

[6] H. Mao, Y. Chen, M. Jaeger, T. D. Nielsen, K. G. Larsen, and B. Nielsen, "Learning Markov decision processes for model checking," in *QFM*, 2012, to appear.

[7] H. Hermanns and L. Zhang, "From concurrency models to numbers - performance and dependability," in *Software and Systems Safety - Specification and Verification*, pp. 182–210, 2011.

[8] B. Settles, *Active Learning*. Morgan & Claypool, 2012.

[9] S. Tong, *Active learning: theory and application*. PhD thesis, Stanford University, 2001.

[10] B. Anderson and A. Moore, "Active learning for hidden Markov models: Objective functions and algorithms," in *ICML*, pp. 9–16, 2005.

[11] B. Steffen, F. Howar, and M. Merten, "Introduction to active automata learning from a practical perspective," in *Formal Methods for Eternal Networked Software Systems*, vol. 6659, pp. 256–296, 2011.

[12] J. Bongard and H. Lipson, "Active coevolutionary learning of deterministic finite automata," *Journal Machine Learning Research.*, vol. 6, pp. 1651–1678, Dec. 2005.

[13] C. Baier and J.-P. Katoen, *Principles of model checking*. MIT Press, 2008.

[14] A. Pnueli, "The temporal logic of programs," in *FOCS*, 1977.

[15] H. Wassily, "Probability inequalities for sums of bounded random variables," *Journal of the American Statistical Association*, vol. 58, no. 58, pp. 13–30, 1963.

[16] M. H. DeGroot, *Optimal Statistical Decisions*. Wiley-Interscience, 2004.

[17] S. Tong and D. Koller, "Active learning for parameter estimation in Bayesian networks," in *NIPS*, pp. 647–653, 2000.

[18] D. N. Jansen, "Probabilistic UML statecharts for specification and verification a case study," in *Critical Systems Development with UML – Proc. of the UML'02 workshop*, pp. 121–132, 2002.

[19] M. Kwiatkowska, G. Norman, and D. Parker, "PRISM 4.0: Verification of probabilistic real-time systems," in *CAV*, pp. 585–591, 2011.